1 The Monkey Banana program

Below is the program as shown in Figure 2.14 [1].

% Figure 2.14 A program for the monkey and banana problem.
% move( State1, Move, State2): making Move in State1 results in State2;
% a state is represented by a term:
% state( MonkeyHorizontal, MonkeyVertical, BoxPosition, HasBanana)

% clause a.
move( state( middle, onbox, middle, hasnot), % Before move
    grasp, % Grasp banana
    state( middle, onbox, middle, has) ). % After move

% clause b.
move( state( P, onfloor, P, H),
    climb, % Climb box
    state( P, onbox, P, H )).

% clause c.
move( state( P1, onfloor, P1, H),
    push( P1, P2), % Push box from P1 to P2
    state( P2, onfloor, P2, H )).

% clause d.
move( state( P1, onfloor, B, H),
    walk( P1, P2), % Walk from P1 to P2
    state( P2, onfloor, B, H )).

% canget( State): monkey can get banana in State

% clause e.
canget( state( _, _, _, has )). % can 1: Monkey already has it

% clause f.
canget( State1) :-
    move( State1, Move, State2), % change from state1 to state2
    canget( State2). % Can we get the banana in the new state?
2 A sample session

The following is a sample query, saying that at the very beginning, a monkey is at the door, on the floor, without the banana, while the box is in the middle. The query is that with such an initial configuration, can the monkey get the banana?

% a sample query
?- canget(state(atdoor, onfloor, atwindow, hasnot)).

If you send in this query, the Prolog will say ‘yes’. To find out the details, we can use the tracing facility. Below shows the original process when trace is turned on.

Welcome to SWI-Prolog (Multi-threaded, Version 5.6.32)
Copyright (c) 1990-2007 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

% c:/Archive/Courses/CS350/CodesForTheBook/monkeyAndBanana.pl compiled 0.02 sec, 1,880 bytes
1 ?- trace.

Yes
[trace] 1 ?- canget(state(atdoor, onfloor, atwindow, hasnot)).
  Call: (7) canget(state(atdoor, onfloor, atwindow, hasnot)) ? creep
  Call: (8) move(state(atdoor, onfloor, atwindow, hasnot), _L186, _L169) ? creep
  Exit: (8) move(state(atdoor, onfloor, atwindow, hasnot), walk(atdoor, _G715),
           state(_G715, onfloor, atwindow, hasnot)) ? creep
  Call: (8) canget(state(_G715, onfloor, atwindow, hasnot)) ? creep
  Call: (9) move(state(_G715, onfloor, atwindow, hasnot), _L203, _L186) ? creep
  Exit: (9) move(state(atwindow, onfloor, atwindow, hasnot), climb,
           state(atwindow, onbox, atwindow, hasnot)) ? creep
  Call: (9) canget(state(atwindow, onbox, atwindow, hasnot)) ? creep
  Call: (10) move(state(atwindow, onbox, atwindow, hasnot), _L240, _L223) ? creep
  Fail: (10) move(state(atwindow, onbox, atwindow, hasnot), _L240, _L223) ? creep
  Fail: (9) canget(state(atwindow, onbox, atwindow, hasnot)) ? creep
  Redo: (9) move(state(_G715, onfloor, atwindow, hasnot), _L203, _L186) ? creep
  Exit: (9) move(state(atwindow, onfloor, atwindow, hasnot), push(atwindow, _G723),
           state(_G723, onfloor, _G723, hasnot)) ? creep
  Call: (9) canget(state(_G723, onfloor, _G723, hasnot)) ? creep
  Call: (10) move(state(_G723, onfloor, _G723, hasnot), _L241, _L224) ? creep
  Exit: (10) move(state(_G723, onfloor, _G723, hasnot), climb,
           state(_G723, onbox, _G723, hasnot)) ? creep
  Call: (10) canget(state(_G723, onbox, _G723, hasnot)) ? creep
  Call: (11) move(state(_G723, onbox, _G723, hasnot), _L278, _L261) ? creep
  Exit: (11) move(state(middle, onbox, middle, hasnot), grasp,
           state(middle, onbox, middle, has)) ? creep
  Call: (11) canget(state(middle, onbox, middle, has)) ? creep
  Exit: (11) canget(state(middle, onbox, middle, has)) ? creep
  Exit: (10) canget(state(middle, onbox, middle, hasnot)) ? creep
  Exit: (9) canget(state(middle, onfloor, middle, hasnot)) ? creep
3 An annotated version of the tracing session

We now explain all the steps in the above process:

1 ?- trace. % Turn on the trace.
Yes % trace is now on

% Enter the query
[trace] 1 ?- canget(state(atdoor, onfloor, atwindow, hasnot)).

% 1. Clause e does not work
Call: (7) canget(state(atdoor, onfloor, atwindow, hasnot)) ? creep

% 2. Try clause f
Call: (8) move(state(atdoor, onfloor, atwindow, hasnot), _L186, _L169) ? creep

% 3. Since atdoor does not match with middle, clause a can’t be applied;
% since atdoor is not the same as atwindow, neither clause b nor c
% can be applied. Thus, only clause d can be applied.
Exit: (8) move(state(atdoor, onfloor, atwindow, hasnot), walk(atdoor, _G715),
state(_G715, onfloor, atwindow, hasnot)) ? creep

% 4. Now, continue with the second part of clause f in this case.
Call: (8) canget(state(_G715, onfloor, atwindow, hasnot)) ? creep

% 5. Again, clause e does not work, we have to try clause f.
Call: (9) move(state(_G715, onfloor, atwindow, hasnot), _L203, _L186) ? creep

% 6. Since the box is atwindow, but not middle, clause a can’t be applied,
% but clauses b, c and d can all be applied, since _G715
% is a variable, and can be instantiated with ‘atwindow’. We choose to
% apply clause b first, with variable _G715 being instantiated
% to at window.
Exit: (9) move(state(atwindow, onfloor, atwindow, hasnot), climb,
state(atwindow, onbox, atwindow, hasnot)) ? creep

% 7. We now continue with the second part of clause f, trying to see if it
% works in the new state.
Call: (9) canget(state(atwindow, onbox, atwindow, hasnot)) ? creep
% 8. Again it does not work with clause e, because of the 'hasnot', so we try clause f.

Call: (10) move(state(atwindow, onbox, atwindow, hasnot), _L240, _L223) ? creep

% 9. Now, for this goal, nothing matches: since the box is 'atwindow',
% clause a fails to apply; since the monkey is now 'onbox', so neither
% clause b, c, nor d applies. Thus, this query fails.

Fail: (10) move(state(atwindow, onbox, atwindow, hasnot), _L240, _L223) ? creep

% 10. So is the original query as formed in Step 6, when we selected clause b.

Fail: (9) canget(state(atwindow, onbox, atwindow, hasnot)) ? creep

% 11. We now have to make an alternative choice to resolve the query as formed in Step 6.

Redo: (9) move(state(_G715, onfloor, atwindow, hasnot), _L203, _L186) ? creep

% 12. We apply clause c instead, by instantiating _G715 with 'atwindow' again.
% We are not sure where to push to, thus _G723 is not instantiated.

Exit: (9) move(state(atwindow, onfloor, atwindow, hasnot), push(atwindow, _G723),
state(_G723, onfloor, _G723, hasnot)) ? creep

% 13. We try to resolve the second part of the query to see if the monkey can
% get it in the new state. Clause e is not applicable because of the 'hasnot'.

Call: (9) canget(state(_G723, onfloor, _G723, hasnot)) ? creep

% 14. Again, we have to try clause f, try to make a move.

Call: (10) move(state(_G723, onfloor, _G723, hasnot), _L241, _L224) ? creep

% 15. Since the monkey is 'onfloor', clause a is not applicable, but the next one,
% clause b is, since the positions of the monkey and the box
% is the same. As a result, the money is not 'onbox'.

Exit: (10) move(state(_G723, onfloor, _G723, hasnot), climb,
state(_G723, onbox, _G723, hasnot)) ? creep

% 16. We apply clause b, then try to see if the monkey can get it in the new state
% when it is in position G_723. Again, clause e cannot be applied.

Call: (10) canget(state(_G723, onbox, _G723, hasnot)) ? creep

% 17. We try clause f, when we have to decide which movement rule is made use of.

Call: (11) move(state(_G723, onbox, _G723, hasnot), _L278, _L261) ? creep

% 18. In this case, since the monkey is now 'onbox', clause a can be applied,
% when _G723, introduced in Step 12 is now finally instantiated with
the value of 'middle'. This leads to a new state when the monkey has 'grasp'ed
the banana.

Exit: (11) move(state(middle, onbox, middle, hasnot), grasp,
state(middle, onbox, middle, has)) ? creep

% 19. Let's see if the monkey has got it in this new state.

Call: (11) canget(state(middle, onbox, middle, has)) ? creep

% 20. Clause e now applies, the query succeeds.

Exit: (11) canget(state(middle, onbox, middle, has)) ? creep

% 21. Now the query formed in Step 16 succeeds, with _G723 instantiated with 'middle'.

Exit: (10) canget(state(middle, onbox, middle, hasnot)) ? creep

% 22. Thus, the query formed in Step 13 succeeds.

Exit: (9) canget(state(middle, onfloor, middle, hasnot)) ? creep

% 23. The query formed in Step 4 succeeds, with _G715 instantiated with 'atwindow',
% done in Step 12.

Exit: (8) canget(state(atwindow, onfloor, atwindow, hasnot)) ? creep

% 24. Finally, the original query succeeds.

Exit: (7) canget(state(atdoor, onfloor, atwindow, hasnot)) ? creep

% 25. It is now done.

Yes

4 What should the monkey do?

From the above annotated tracing results, we find out that the choices made in the sequence of Steps 3, 12, 15, and 18 lead to the success achieved at Step 19, i.e., if the monkey starts at the door, on floor, without getting the banana yet, and the box is at the window, the monkey should take the following sequence of actions to get the banana:

1. The monkey walks from the door place to the window (Step 3).
2. It then pushes the box from the window place to the middle place (Step 12).
3. The monkey then climbs onto the box (Step 15);
4. and finally, grasp the banana (Step 18).
References